Nom: GUIZANI **Prénom:** Ahmed **class :** 5A-SWE-Groupe 2

Question 1 :

On commence par transférer le fichier texte vers la machine virtuelle, grâce à la commande *scp*.

```
C:\Users\pheup>scp "D:\Users\pheup\Documents\ESIEA\5A\Big data architecture and data
 processing\Labs\Ressources LAB1\q1_text.txt" training@192.168.1.19:/home/training
training@192.168.1.19's password:
q1_text.txt
  100%  202   202.1KB/s   00:00

C:\Users\pheup>
```

Ensuite, depuis la machine virtuelle, on place ce fichier dans notre répertoire d'entrée dans hdfs. Pour cela, on utilise la commande suivante.

```
$ hdfs dfs -put /home/training/Materials_Lab1/q1_text.txt /home/training/lab1/input
```

On exécute ensuite le programme wordcount.jar grâce à Hadoop, en spécifiant notre fichier d'entrée, et notre répertoire de sortie.

```
[training@linux demo]$ hadoop jar wordcount.jar demo.WCount /home/training/lab1/input/q1_text.txt /home/training/lab1/output/
1
21/09/26 16:09:24 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement To
ol for the same.
21/09/26 16:09:25 INFO input.FileInputFormat: Total input paths to process : 1
21/09/26 16:09:25 WARN snappy.LoadSnappy: Snappy native library is available
21/09/26 16:09:25 INFO snappy.LoadSnappy: Snappy native library loaded
21/09/26 16:09:25 INFO mapred.JobClient: Running job: job_202109261448_0009
21/09/26 16:09:26 INFO mapred.JobClient:  map 0% reduce 0%
21/09/26 16:09:30 INFO mapred.JobClient:  map 100% reduce 0%
21/09/26 16:09:33 INFO mapred.JobClient:  map 100% reduce 100%
21/09/26 16:09:33 INFO mapred.JobClient: Job complete: job_202109261448_0009
```

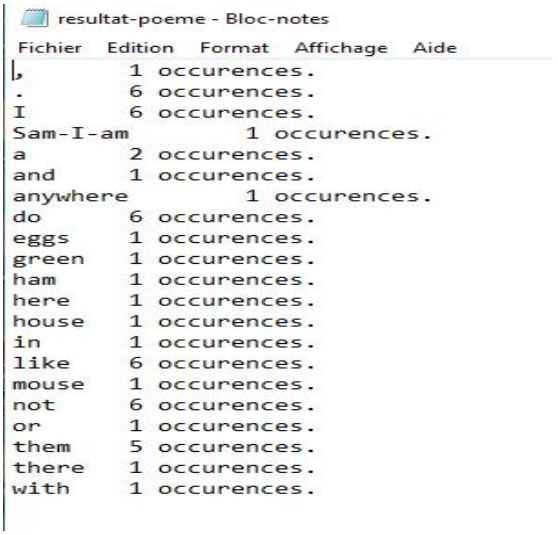
A la fin de d'exécution, on peut directement lire le résultat dans le terminal, par exemple grâce à la commande *cat*.

```
[training@linux demo]$ hdfs dfs -cat /home/training/lab1/output/1/part-r-00000
,       1 occurences.
.       6 occurences.
I       6 occurences.
Sam-I-am        1 occurences.
a       2 occurences.
and     1 occurences.
anywhere        1 occurences.
do      6 occurences.
eggs    1 occurences.
green   1 occurences.
ham     1 occurences.
here    1 occurences.
house   1 occurences.
in      1 occurences.
like    6 occurences.
mouse   1 occurences.
not     6 occurences.
or      1 occurences.
them    5 occurences.
there   1 occurences.
with    1 occurences.
[training@linux demo]$
```

On peut aussi transférer le fichier résultant vers la machine hôte, toujours en utilisant la commande *scp*.

```
[training@localhost bin]$ scp /home/training/resultat-poeme.txt saief2@192.168.56.1:resultat-poeme.txt
saief2@192.168.56.1's password:
resultat-poeme.txt                                            100%  390     0.4KB/s   00:00
[training@localhost bin]$
```

Le fichier est ensuite lisible dans n'importe quel éditeur.



Question 2 : Anagrams

Pseudocode:

Map:

Input: key = LongWritable, value = Text

Output: Key = Text, value = Text

- Remove the spaces from the line ;
- Create a string value made of the characters from the line, ordered alphabetically. We will call this string value 'line_sorted'. For example, 'cat' and 'tac' would be transformed into 'act' ;
- Add an output key/value pair of <line_sorted, line>, that is the line sorted alphabetically, and the line itself, without the spaces.

Reduce:

Input: key = Text, value = Text

Output: key = Text, value = Text

- Join the anagrams together as they have become equals in the sorted array.
- Add a key/value output, which are the text word and its anagrams separated by ','

Implémentation:

Vous trouverez le code Java et le fichier jar ci-joint avec le rendu.
Nous avons commencé par créer le fichier jar à partir de nos driver, mapper et reducer.

```
[training@localhost ~]$ cd /home/training/workspace
[training@localhost workspace]$ ls
averagewordlength   Lab1               partitioner         wordcount
combiner            Lab1-anagrams      test                writables
counters            Lab1-BigData       training
createsequencefile  log_file_analysis  WCount-lab1
inverted_index      mrunit             word_co-occurrence
[training@localhost workspace]$ cd Lab1-anagrams
[training@localhost Lab1-anagrams]$ ls
bin  src
[training@localhost Lab1-anagrams]$ cd bin
[training@localhost bin]$ ls
anagrams
[training@localhost bin]$ cd anagrams
[training@localhost anagrams]$ ls
Anagram$AnagramMapper.class  Anagram$AnagramReducer.class  Anagram.class
[training@localhost anagrams]$ cd ../
[training@localhost bin]$ jar -cvf anagrams.jar anagrams
added manifest
adding: anagrams/(in = 0) (out= 0)(stored 0%)
adding: anagrams/Anagram.class(in = 2431) (out= 1135)(deflated 53%)
adding: anagrams/Anagram$AnagramMapper.class(in = 2407) (out= 963)(deflated 59%)
adding: anagrams/Anagram$AnagramReducer.class(in = 2912) (out= 1138)(deflated 60
%)
[training@localhost bin]$
```

Ensuite, nous avons créé nos répertoires d'entrée (input) et de (sortie), puis exécuté le programme grâce à Hadoop.

```
[training@localhost bin]$ hadoop fs -mkdir /home/training/lab2
[training@localhost bin]$ hadoop fs -mkdir /home/training/lab2/input
[training@localhost bin]$ hadoop fs -mkdir /home/training/lab2/output
[training@localhost bin]$ hadoop fs -put /home/training/Q2test.txt  /home/training/lab2/input/Q2test.txt
[training@localhost bin]$ hadoop fs -cat /home/training/lab2/input/Q2test.txt
captain over rome
lives
cat
banana
emperor octavian
act
tac
elvis

[training@localhost bin]$ hadoop jar anagrams.jar anagrams.Anagram /home/trainin
g/lab2/input /home/training/lab2/output/out
21/09/24 00:17:13 WARN mapred.JobClient: Use GenericOptionsParser for parsing th
e arguments. Applications should implement Tool for the same.
21/09/24 00:17:14 INFO input.FileInputFormat: Total input paths to process : 1
21/09/24 00:17:14 WARN snappy.LoadSnappy: Snappy native library is available
21/09/24 00:17:14 INFO snappy.LoadSnappy: Snappy native library loaded
21/09/24 00:17:14 INFO mapred.JobClient: Running job: job_202109232230_0002
21/09/24 00:17:15 INFO mapred.JobClient:  map 0% reduce 0%
21/09/24 00:17:23 INFO mapred.JobClient:  map 100% reduce 0%
21/09/24 00:17:27 INFO mapred.JobClient:  map 100% reduce 100%
21/09/24 00:17:29 INFO mapred.JobClient: Job complete: job_202109232230_0002
21/09/24 00:17:29 INFO mapred.JobClient: Counters: 32
21/09/24 00:17:29 INFO mapred.JobClient:   File System Counters
21/09/24 00:17:29 INFO mapred.JobClient:     FILE: Number of bytes read=159
21/09/24 00:17:29 INFO mapred.JobClient:     FILE: Number of bytes written=362104
21/09/24 00:17:29 INFO mapred.JobClient:     FILE: Number of read operations=0
21/09/24 00:17:29 INFO mapred.JobClient:     FILE: Number of large read operations=0
21/09/24 00:17:29 INFO mapred.JobClient:     FILE: Number of write operations=0
21/09/24 00:17:29 INFO mapred.JobClient:     HDFS: Number of bytes read=188
21/09/24 00:17:29 INFO mapred.JobClient:     HDFS: Number of bytes written=85
21/09/24 00:17:29 INFO mapred.JobClient:     HDFS: Number of read operations=2
21/09/24 00:17:29 INFO mapred.JobClient:     HDFS: Number of large read operations=0
21/09/24 00:17:29 INFO mapred.JobClient:     HDFS: Number of write operations=1
21/09/24 00:17:29 INFO mapred.JobClient:   Job Counters
21/09/24 00:17:29 INFO mapred.JobClient:     Launched map tasks=1
21/09/24 00:17:29 INFO mapred.JobClient:     Launched reduce tasks=1
21/09/24 00:17:29 INFO mapred.JobClient:     Data-local map tasks=1
21/09/24 00:17:29 INFO mapred.JobClient:     Total time spent by all maps in occupied slots (ms)=9254
21/09/24 00:17:29 INFO mapred.JobClient:     Total time spent by all reduces in occupied slots (ms)=4470
21/09/24 00:17:29 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
21/09/24 00:17:29 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
21/09/24 00:17:29 INFO mapred.JobClient:   Map-Reduce Framework
21/09/24 00:17:29 INFO mapred.JobClient:     Map input records=10
21/09/24 00:17:29 INFO mapred.JobClient:     Map output records=10
21/09/24 00:17:29 INFO mapred.JobClient:     Map output bytes=133
21/09/24 00:17:29 INFO mapred.JobClient:     Input split bytes=120
```

Nous avons pu visualiser le fichier de sortie directement dans le terminal.

```
21/09/24 00:17:29 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
21/09/24 00:17:29 INFO mapred.JobClient:   Map-Reduce Framework
21/09/24 00:17:29 INFO mapred.JobClient:     Map input records=10
21/09/24 00:17:29 INFO mapred.JobClient:     Map output records=10
21/09/24 00:17:29 INFO mapred.JobClient:     Map output bytes=133
21/09/24 00:17:29 INFO mapred.JobClient:     Input split bytes=120
21/09/24 00:17:29 INFO mapred.JobClient:     Combine input records=0
21/09/24 00:17:29 INFO mapred.JobClient:     Combine output records=0
21/09/24 00:17:29 INFO mapred.JobClient:     Reduce input groups=5
21/09/24 00:17:29 INFO mapred.JobClient:     Reduce shuffle bytes=159
21/09/24 00:17:29 INFO mapred.JobClient:     Reduce input records=10
21/09/24 00:17:29 INFO mapred.JobClient:     Reduce output records=3
21/09/24 00:17:29 INFO mapred.JobClient:     Spilled Records=20
21/09/24 00:17:29 INFO mapred.JobClient:     CPU time spent (ms)=2090
21/09/24 00:17:29 INFO mapred.JobClient:     Physical memory (bytes) snapshot=275230720
21/09/24 00:17:29 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=810835968
21/09/24 00:17:29 INFO mapred.JobClient:     Total committed heap usage (bytes)=223805440
[training@localhost bin]$ hadoop fs -ls /home/training/lab2/output/out/
Found 3 items
-rw-r--r--   1 training supergroup          0 2021-09-24 00:17 /home/training/lab2/output/out/_SUCCESS
drwxr-xr-x   - training supergroup          0 2021-09-24 00:17 /home/training/lab2/output/out/_logs
-rw-r--r--   1 training supergroup         85 2021-09-24 00:17 /home/training/lab2/output/out/part-r-00000
[training@localhost bin]$ hadoop fs -cat /home/training/lab2/output/out/part-r-00000
aaceeimnooprrtv captain over rome,emperor octavian
act     cat,act,tac
eilsv   lives,elvis
[training@localhost bin]$ hadoop fs -get /home/training/lab2/output/out/part-r-00000 /home/training/outputAnagrams.txt
[training@localhost bin]$
```

Question 3 : K-NN (almost)

Map:

Input: key = LongWritable, value = Text

Output: key = DoubleWritable, value = Text

- Initialize a test example called *test*, by creating an instance of the class *Example* ;
- Turn the line into an instance of the class *Example*, called it *training_example*, using the static method *Example.readExample(line)* ;
- Calculate the distance from the current example to the test example, by using *getDistance(training_example, test)* ;
- Add an output key/value pair of <*distance*, *training_example.class_name*>, that is the distance to the test example, and the class name of the current training example (if we were processing iris flowers, it could be the variety of this flower).

Reduce:

Input: key = DoubleWritable, value = Text

Output: key = Text, value = DoubleWritable

- We have nothing to do, as we already have our pairs of label/distance, that have been automatically sorted by distance. We simply return the input data, while switching the key and the value, as requested in the LAB ;
- Add an output key/value pair of <*training_example.class_name, distance*>, that is the class name of the current training example, and the distance to the test example.

Question 4 : Feature normalization

- Initialize a key/value list called *min_max*, accessible both by the mapper and the reducer. It will store the minimum and maximum values of each feature of the dataset. key = feature name, value = array of two values (minimum and maximum, respectively initialized to MAX_DOUBLE and MIN_DOUBLE).

Map:

Input: key = Text, value = Text

Output: key = Text, value = DoubleWritable

- Parse the line and create an instance of *Example*, giving access to the features' names and values of this line ;
- For each feature *feature_name* of this *Example:*
    - if *example[feature_name]* < *min_max[feature_name][0]*
        - store this new minimum in *min_max[feature_name][0]*
    - if *example[feature_name]* > *min_max[feature_name][1]*
        - store this new maximum in *min_max[feature_name][1]*
- Add an output key/value pair of <*Example.label*, *Example.values*>

Reduce:

Input: key = Text, value = DoubleWritable

Output: key = Text, value = DoubleWritable

- Parse the line and create an instance of *Example*, giving access to the features' names and values of this line ;
- For each feature *feature_name* of this *Example:*
    - *min = min_max[feature_name][0]*
    - *max = min_max[feature_name][1]*
    - *example[feature_name] = (example[feature_name] - min) / (max - min)*
- Add an output key/value pair of <*Example.label*, *Example.values*>, that is the label of this example, and its values.